DESAIN DAN IMPLEMENTASI ROBOT MOBIL OTOMATIS PENGHINDAR HAMBATAN BERBASIS SENSOR KINECT: PENGENALAN KONFIGURASI LINTASAN MENGGUNAKAN JARINGAN SYARAF TIRUAN

DESIGN AND IMPLEMENTATION OF OBSTACLE AVOIDANCE AUTONOMOUS MOBILE ROBOT BASED ON KINECT SENSOR : IDENTIFICATION OF PATH CONFIGURATION USING NEURAL NETWORK

Alfredo Rianto[1], Agus Virgono, Ir., MT.[2], Umar Ali Ahmad, S.T, M.T.[3]

[1,2,3]Prodi S1 Sistem Komputer, Fakultas Teknik Elektro, Universitas Telkom University
[1]alfredorianto@gmail.com, [2]avirgono@telkomuniversity.ac.id, [3]umar@telkomuniversity.ac.id

Abstrak

Banyak mobile robot yang dikembangkan memiliki sistem navigasi sehingga dapat mengenali dan berinteraksi dengan lingkungan. Kemampuan mobile robot dapat menavigasikan lintasan secara otomatis dapat membawa robot dengan aman sampai tujuan. Dengan kemampuan seperti demikian, akan lebih baik jika mobile robot tersebut dilengkapi kemampuan menghindari setiap benda lain yang ada di sekitar robot. Mobile robot otomatis dapat dikatakan berhasil dalam navigasinya, jika robot tersebut dapat memperoleh informasi berupa data jarak dan sudut dari kedua sisi benda yang berada di lingkungan. Dengan informasi demikian, maka robot secara otomatis dapat menentukan arah dan tujuan selanjutnya sesuai dengan kondisi dari lingkungan.

Pada tugas akhir ini dibahas tentang pembuatan sistem kendali pintar pada Mobile robot otomatis berbasis kinect menggunakan Jaringan Syaraf Tiruan. Jaringan Saraf Tiruan berfungsi mengirim sinyal derajat 0-180 secara serial ke sistem kendali fuzzy sebagai inputan fuzzy untuk menggerakan robot menuju tujuan tanpa campur tangan manusia dalam suatu lingkungan yang telah disesuaikan. Topologi sistem Jaringan Syaraf Tiruan yang digunakan adalah jenis jaringan syaraf multilayer feedforward yang dilatih dengan algoritma gradian turun backpropagation, serta topologi sistem Jaringan Syaraf Tiruan ini dibangun menggunakan aplikasi Visual Studio 2013 dan dengan menggunakan library yang dirancang sebelumnya sesuai dengan kebutuhan dalam mengenali konfigurasi objek yang telah disesuaikan.

Metode yang diterapkan pada Aplikasi dalam Tugas Akhir ini adalah jaringan syaraf multilayer feedforward yang dilatih dengan algoritma gradian turun backpropagation pada robot dengan nilai variabel yang baik saat Error threshold = 0,01; Total Epoch = 200000, 20000, dan 180000; Learning rate = 0.08, 0.008, 0.04, dan 0.009; Hidden layer = 0.

Kata Kunci : Mobile Robot, sistem kendali pintar, jaringan syaraf tiruan, multilayer feedforward, backpropagation

Abstract

There are many mobile robots which are developed to have navigation system so that they can recognize and interact with their environment. The ability of mobile robots can navigate automatically the tracks which can carry the robots to the destination safely. With such capabilities, the mobile robots would be better if equipped with the ability to avoid any other objects that exist around the robot. The mobile robots are automatically said to be successful in navigation, when the robots can obtain detailed information in the form of distance and angle data from both sides of the objects in the environment. With such information, the robots can automatically determine the direction and the next destination in accordance with the conditions of the environment.

This final project discussed about making smart control systems in automatic mobile robots using kinect based on Neural Networks. Neural Networks have a function to send a signal of angle 0-180 serially to fuzzy control system as fuzzy input to drive the robot to the destination without human intervention in an environment that has been customized. Neural Networks system topology used is the type of multilayer feedforward neural network trained with gradient down backpropagation algorithm, and topology Neural Network system was built using Visual Studio 2013 application and using the library previously designed according to needs in recognizing objects that have been customized configuration.

The method applied to the application in this final project is a multilayer feedforward neural network trained with backpropagation algorithm gradian down with good variable value when the error threshold = 0,01; total epoch = 200000, 20000, dan 180000; learning rate = 0.08, 0.008, 0.04, dan 0.009; hidden layer = 0.

Keywords: mobile robots, intelligent control systems, neural networks, multilayer feedforward, backpropagation

1. Pendahuluan

Perkembangan teknologi mobile robot telah banyak berperan penting dalam masyarakat moderen. Banyak mobile robot yang dikembangkan memiliki sistem navigasi sehingga dapat mengenali dan berinteraksi dengan lingkungan. Kemampuan mobile robot tersebut dapat diimplementasikan pada mobil otomatis yang

dapat menavigasikan lintasan secara otomatis dan dapat membawa robot dengan aman sampai tujuan. Selain itu robot mobil otomatis dapat menghindari setiap benda yang ada di sekitar robot.

Dengan kemampuan seperti demikian, akan lebih baik jika mobile robot tersebut diimplementasikan ke dalam mobil otomatis yang dapat menavigasikan lintasan secara otomatis dan dapat membawa robot dengan aman sampai tujuan sambil menghindari setiap benda lain yang ada di sekitar robot [1].

Mobile robot otomatis dapat dikatakan berhasil dalam navigasinya, jika robot tersebut dapat memperoleh informasi yang rinci tentang lingkungan terdekat dengan robot. Informasi tersebut berupa data jarak dan sudut dari kedua sisi benda yang berada di lingkungan. Dengan informasi demikian, maka robot secara otomatis dapat menentukan arah dan tujuan selanjutnya sesuai dengan kondisi dari lingkungan. Untuk dapat membuat sebuah sistem yang compact, maka diperlukan sebuah sensor yang dapat mendeteksi jarak pada 1 dimensi dan scan objek pada 2D [8]. Kemudian informasi dari hasil deteksi tersebut akan diolah oleh sebuah algoritma yang telah disesuaikan oleh kondisi lingkungan yang memungkinkan untuk dilalui robot. Lalu keputusan yang dikeluarkan oleh algoritma tersebut akan dieksekusi oleh sistem kendali *fuzzy* pada robot agar robot dapat bergerak ke arah yang sesuai [4].

Dari perihal di atas, maka pada Tugas Akhir ini dibahas tentang pembuatan sistem kendali pintar menggunakan jaringan syaraf tiruan yang berfungsi mengirim sinyal derajat 0-180 secara serial ke sistem kendali *fuzzy* sebagai inputan *fuzzy* untuk menggerakan robot menuju tujuan tanpa campur tangan manusia dalam suatu lingkungan yang telah disesuaikan. Topologi sistem jaringan syaraf tiruan yang digunakan adalah jenis jaringan syaraf multilayer feedforward yang dilatih dengan algoritma gradian turun backpropagation [4].

2. Dasar Teori

2.1. *Mobile Robot*

Mobile robot atau robot bergerak adalah konstruksi robot yang karakteristiknya memiliki aktuator berupa roda untuk menggerakan keseluruhaan bagian robot tersebut, sehingga robot dapat berpindah posisi dari satu titik ke titik yang lain. Robot ini mempunyai kemampuan untuk membuat keputusan sendiri, serta memiliki sistem kendali dan catu daya yang saling terintegrasi dan juga mempunyai kemampuan navigasi yaitu sejumlah operasi yang memungkinkan robot mencapai tujuan tertentu [4].

Banyak robot bergerak dengan teknik pengontrolan yang sudah ditanamkan didalamnya, telah diaplikasikan serta digunakan di berbagai tempat pada masyarakat modern seperti untuk pengawasan dan tugas-tugas keamanan dalam ruangan. Untuk pengembangan mobile robot otonom, terdapat beberapa aspek yang harus dipertimbangkan dalam merancangnya antara lain adalah aplikasi, platform dasar robot dan sensor yang diterapkan dalam menyusun hardware robot. Selain itu, komponen yang sangat penting dari robot terdiri dari kontrol dan sistem navigasi, yang merupakan perangkat lunak dari sistem robot tersebut [2].

2.2. Model Backpropagation pada Metode Jaringan Syaraf Tiruan

Proses dari *backpropagation* pada dasarnya sama dengan proses *feedforward* (propagasi maju) yang dimulai dari *input* ke *output*. Sedangkan untuk proses *backpropagation* dimulai dari *output* ke *input*. Penjelasan mengenai hal tersebut dapat dilihat pada keterangan di bawah ini.

- **Algoritma *Backpropagation***

Algoritma pelatihan *backpropagation* sederhana dengan satu layar tersembunyi (*hidden layer*) dan dengan fungsi aktivasi logistic sigmoid, memiliki beberapa tahapan yakni sebagai berikut:

- Inisialisasi bobot.
- Tetapkan : Maksimum *Epoch*, Target *Error*, dan *Learning Rate* (α).
- Inisialisasi : *Epoch* = 0, MSE (tergantung kondisi).
- Kerjakan langkah-langkah berikut selama (*Epoch* < Maksimum *Epoch*) dan (MSE > Target *Error*) :
    1. Epoch = Epoch + 1
    2. Untuk setiap pasangan elemen yang akan dilakukan pembelajaran, kerjakan :

Tahap *Feedforward* :

a. Setiap neuron *input* (X_i, i=1,2,3,…,n) menerima sinyal x_i dan meneruskan sinyal tersebut ke semua neuron pada lapisan yang ada diatasnya (lapisan tersembunyi).

b. Setiap neuron pada lapisan tersembunyi (Z_j, j=1,2,3,…,p) menjumlahkan bobot sinyal input dengan persamaan :

$$z\_in_j = V_{oj} + \sum_{i=1}^{n} x_i v_{ij} \qquad (2.1)$$

Dan untuk mendapatkan sinyal outputnya, gunakan fungsi aktivasi :

$$z_j = f(z\_in_j) \qquad (2.2)$$

c. Setiap neuron pada lapisan *output* (Y_k, k = 1,2,3,…,m) menjumlahkan bobot nilai dari sinyal *input* :

$$y\_in_k = w_{0k} + \sum_{i=1}^{p} z_i w_{jk} \qquad (2.3)$$

Dan gunakan fungsi aktivasi untuk menghitung sinyal *output* :

$$y_k = f(y\_in_k)\text{k} \qquad (2.4)$$

Tahap *Backpropagation* :

a. Pada setiap neuron di lapisan *output* (Y_k, k = 1,2,3,…,m) menerima pola target yang sesuai dengan pola *input* pelatihan, kemudian hitung informasi yang *error* dengan persamaan :
$\delta_k = (t_k - y_k) f'(y\_in_k)$ *(2. 5)*

f' adalah turunan dari fungsi aktivasi.
Kemudian hitung koreksi bobot (yang nantinya akan digunakan untuk memperbaiki nilai w_{jk}) dengan rumus :
$\Delta w_{jk} = \alpha\ \delta_k\ z_j$ *(2. 6)*

Dan hitung koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai w_{0k}) dengan rumus :
$\Delta w_{0k} = \alpha\ \delta_k$ *(2. 7)*

Sekaligus mengirimkan δ_k ke neuron yang ada pada lapisan paling kanan.

b. Pada setiap neuron di lapisan tersembunyi (Z_j, j = 1,2,3,…,p) menjumlahkan sejumlah *input*-nya (dari neuron yang berada pada lapisan di kanannya) :
$\delta\_in_j = \sum_{k=1}^{m} \delta_k\ w_{jk}$ *(2. 8)*

Untuk menghitung informasi *error*, kalikan hasil dari persamaan diatas dengan turunan dari fungsi aktivasinya :
$\delta_j = \delta\_in_j\ f'(z\_in_j)$ *(2. 9)*

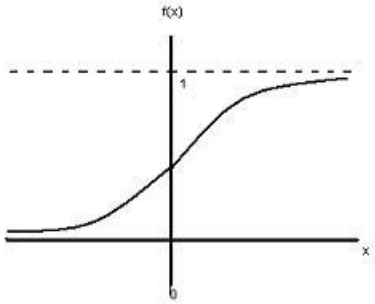
Kemudian hitung nilai koreksi bobot (yang nantinya akan digunakan untuk memperbaiki nilai v_{jk}) dengan rumus :
$\Delta v_{jk} = \alpha\ \delta_j\ x_i$ *(2. 10)*
Kemudian hitung koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai v_{0j}) dengan rumus :
$\Delta v_{0j} = \alpha\ \delta_j$ *(2. 11)*

Tahap perubahan bobot dan bias :

a. Setiap neuron pada lapisan *output* (Y_k, k = 1,2,3,…,m) akan mengalami perubahan bobot dan bias (j=0,1,2,…,p) saat proses *backpropagation* dengan rumus :
$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk}$ *(2. 12)*

Setiap neuron pada lapisan tersembunyi (Z_j, j = 1,2,3,…,p) akan mengalami perubahan bobot dan bias (i=0,1,2,…,n) saat proses *backpropagation* dengan rumus [5] :
$v_{ij}(baru) = v_{ij}(lama) + \Delta v_{ij}$ *(2. 13)*

- **Fungsi Aktivasi**
  Pada umumnya, fungsi aktivasi yang digunakan adalah fungsi sigmoid, dan kemudian mengirimkan sinyal tersebut ke semua neuron pada lapisan *output*.
  Fungsi sigmoid sendiri terdiri menjadi 2, yaitu fungsi sigmoid biner dan fungsi sigmoid bipolar.
  1. Fungsi sigmoid biner digunakan untuk jaringan syaraf yang dilatih dengan menggunakan metode *backpropagation*, dimana fungsi ini sering digunakan pada jaringan syaraf yang nilai keluarannya terletak pada interval 0 sampai 1, ataupun nilai *output* 0 atau 1.

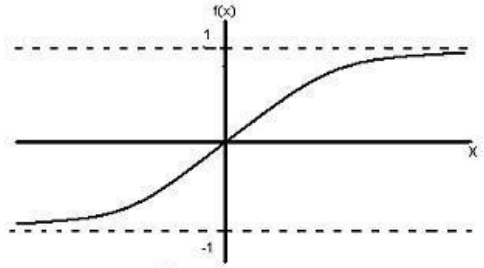


Gambar 2. 2 Ilustrasi fungsi sigmoid biner dengan range (0,1) [5]

Fungsi sigmoid biner dirumuskan sebagai :
$y = f(x) = \frac{1}{1+e^{-\sigma x}}$ *(2. 14)*

Dengan $f'(x) = \sigma f(x)[1 - f(x)]$

2. Fungsi sigmoid bipolar hampir sama kegunaannya dengan fungsi sigmoid biner, hanya saja keluaran dari fungsi ini memiliki interval antara 1 sampai -1.



Gambar 2. 3 Ilustrasi fungsi sigmoid bipolar dengan range (-1,1) [5]

Secara matematis, fungsi sigmoid bipolar dirumuskan sebagai berikut :

$$y = f(x) = \frac{1-e^{-x}}{i+e^{-x}} \qquad (2.\ 15)$$

Dengan : $f'(x) = \frac{\sigma}{2}[1 + f(x)][1 - f(x)]$

Fungsi ini hampir menyerupai dengan fungsi *hyperbolic tangent*. Keduanya memiliki interval yang sama, yaitu antara -1 sampai 1. Secara matematis, fungsi *hyperbolic tangent* dituliskan dengan rumus [5] :

$$y = f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

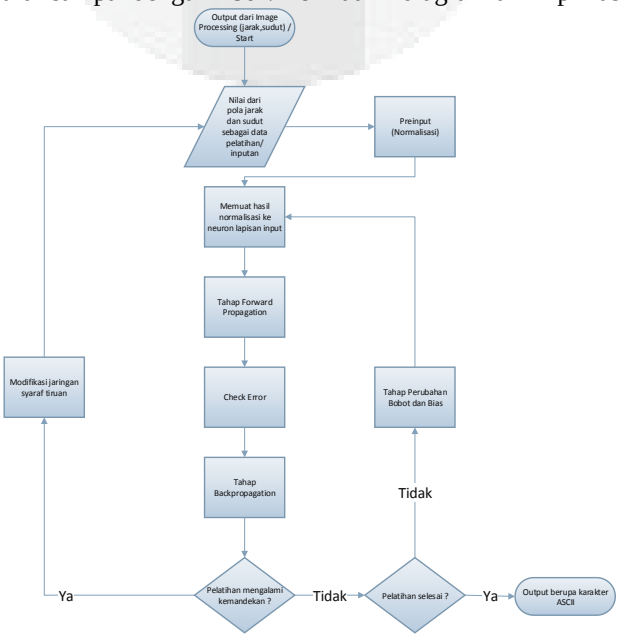
Atau

$$y = f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \qquad (2.\ 16)$$

Dengan : $f'(x) = [1 + f(x)][1 - f(x)]$

3. Perancangan

Pada tugas akhir ini dibuat Aplikasi pada PC untuk mengenali konfigurasi objek dan menghindarinya berdasarkan nilai jarak, nilai sudut α, dan nilai sudut β dari objek yang telah dinormalisasi dan memberikan hasil keputusan pergerakan robot yang diinginkan melalui pelatihan supervised menggunakan komunikasi serial ke mikrokontroler. Pengolahan data pada Aplikasi berupa perhitungan sekumpulan nilai dari jarak, sudut α dan sudut β dengan batasan jarak pembacaan objek dari 50 cm sampai dengan 100 cm di depan sensor Kinect dan vertical tilt range ±27° pada sisi kiri dan kanan sensor Kinect, untuk mendapatkan nilai bobot yang efisien dalam memberikan hasil keputusan untuk pergerakan robot. Proses pengenalan pada robot akan dibuat secara realtime, yaitu ketika kondisi robot bergerak dan mendapatkan masukan dari algoritma multilayer feedforward berupa keputusan untuk berbelok antara 0° sampai dengan 180°. Berikut ini diagram alir Aplikasi :



Gambar 3. 1 Diagram alir Aplikasi

3.1 Normalisasi Data

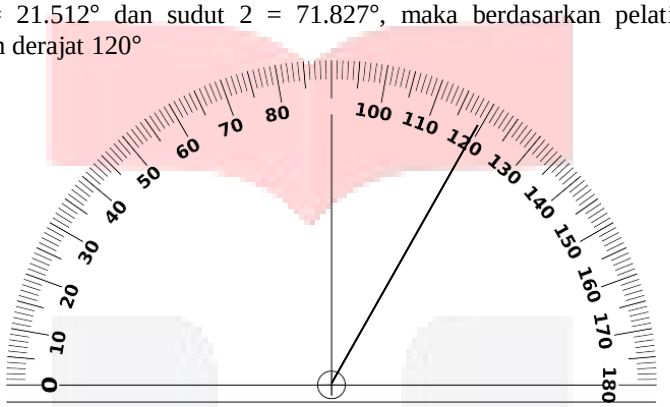
Data file.CSV yang akan diolah oleh aplikasi untuk tahap pelatihan memiliki rentang data dari 400-1000 untuk *distance* (dalam satuan mm) dan rentang data dari 10-166 untuk *angle* (dalam satuan *degree*). Oleh karena itu, untuk memenuhi syarat fungsi aktivasi sigmoid biner agar data tersebut dapat diolah dalam algoritma jaringan syaraf tiruan, maka dilakukan normalisasi nilai dengan persamaan berikut:

$$dist_n = \frac{(dist - 400)}{600} \quad (3.1)$$

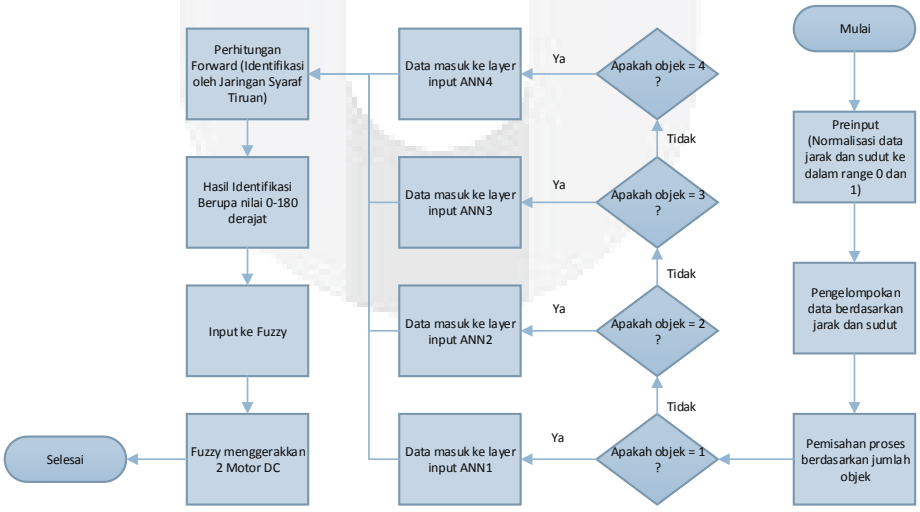
$$ang_n = \frac{(ang - 10)}{156} \quad (3.2)$$

3.2 Algoritma Penyelesaian Masalah pada Automatic Mobile Robot Obstacle Avoidance

Algoritma penyelesaian masalah yang pertama dapat dilihat pada Gambar 3.2. Setelah dimulai robot mendefinisikan suatu variabel belok 0-180 derajat dengan rentang nilai antara 0 dan 1. Pada permulaan robot di set untuk berjalan lurus. Robot akan menentukan arak belok jika menemukan sebuah objek atau dinding yang jaraknya kurang dari 600 mm dari sensor Kinect. Keputusan berbelok ke arah derajat tertentu ditentukan oleh hasil pelatihan *supervised* yang menghasilkan nilai bobot yang optimum dalam pengambilan keputusan. Sebagai contoh, ketika sensor Kinect mendeteksi adanya 1 objek yang terletak di sebelah kiri dari sensor Kinect dengan jarak 484 mm, sudut 1 = 21.512° dan sudut 2 = 71.827°, maka berdasarkan pelatihan *supervised* robot seharusnya bergerak ke arah derajat 120°



Gambar 3.2 Arah derajat 120°



Gambar 3. 3 Algoritma dalam menghindari objek

3.3 Mendapatkan Data Nilai dari jarak, sudut α dan sudut β dari objek yang ditangkap Kinect

Aplikasi ini dirancang untuk mengenali konfigurasi lingkungan yang memungkinkan untuk dilewati oleh *mobile robot* melalui sekumpulan informasi data jarak dan sudut dari 1 objek hingga 4 objek pada pada *range* 50 – 100 cm dari depan sensor Kinect dengan *viewing angle* 43° vertikal dengan 57° bidang pandang horisontal dan *vertical tilt range* $\pm$27° pada sisi kiri dan kanan sensor Kinect yang dilatih pada algoritma *backpropagation,* melalui data latih berbentuk file.CSV yang berisikan nilai jarak dan sudut hasil *capture* dari Kinect dan pengujian pada robot menggunakan algoritma *multilayer feedforward,* yang nantinya akan menggunakan bobot JST hasil pelatihan yang paling efisien dari hasil beberapa pelatihan berbagai kondisi objek.

Tabel 3. 1 Contoh data latih JST menggunakan 4 benda

Dist 1	Alpha 1	Beta 1	Dist 2	Alpha 2	Beta 2	Dist 3	Alpha 3	Beta 3	Dist 4	Alpha 4	Beta4	Outcome
677	84.256	99.64	555	39.523	79.745	649	22.109	70.203	578	135.303	120.385	20
533	85.119	101.69	555	71.32	81.307	537	19.966	70.811	579	135	161.686	170
565	159.941	164.201	559	24.184	39.184	553	18.057	61.201	568	104.533	120.329	180
564	159.547	163.69	554	67.471	83.39	532	18.553	66.809	568	106.077	120.801	10
551	115.414	159.156	553	68.959	80.278	532	17.488	65.53	560	95.412	111.866	150
550	150.401	158.199	553	34.611	81.021	532	18.144	66.709	549	120.774	114.65	150
553	34.237	64.855	554	155.095	161.245	559	16.24	24.484	548	99.94	114.199	160
553	34.716	82.287	555	158.325	163.747	558	18.219	24.444	549	99.645	115.44	170
573	161.192	163.183	552	68.074	80.278	559	18.326	24.444	549	100.305	113.291	180

Aplikasi yang dirancang hanya dibatasi maksimal mendeteksi objek hingga 4 benda karena dengan 4 benda sudah hampir memenuhi *viewing angle* dari Kinect. Pada table terlihat bahwa sebagian besar nilai pada *Distance* 1 sampai *Distance* 4 berada pada *range* 530-580 mm karena pada jarak tersebut merupakan jarak ideal bagi robot untuk mengambil keputusan untuk berbelok dalam menghindari objek.

4. Pengujian

4.1 Pengujian Blackbox

Pengujian ini bertujuan untuk mengetahui konfigurasi yang optimum dengan memperhatikan parameter-parameter seperti error, LR, dan epoch.

	ANN konfigurasi			jumlah hidden layer	Hasil Training							
					ANN 1		ANN 2		ANN 3		ANN 4	
percobaan	Error threshold	Epoch	Learning rate		Error	Epoch	Error	Epoch	Error	Epoch	Error	Epoch
1	0.01	20000	0.04	0	16.50202	20000	17.5031	20000	7.519698	20000	7.752559	20000
2	0.005	20000	0.04	0	16.50202	20000	17.5031	20000	8.337114	20000	6.546557	20000
3	0.01	200000	0.08	0	16.50199	200000	10.33757	200000	9.123058	200000	9.170022	200000
4	0.01	200000	0.008	0	16.50122	200000	4.862313	200000	8.537376	200000	5.334186	200000
5	0.01	20000	0.04	25	16.50599	20000	17.50071	20000	9.940909	20000	9.066319	20000
6	0.01	20000	0.04	100	16.51182	20000	17.51074	20000	8.973921	20000	7.244238	20000
7	0.01	180000	0.009	0	16.50129	180000	17.50048	180000	6.899268	180000	5.415663	180000
8	0.01	300000	0.008	0	16.50934	300000	16.92864	300000	7.554969	300000	7.370777	300000

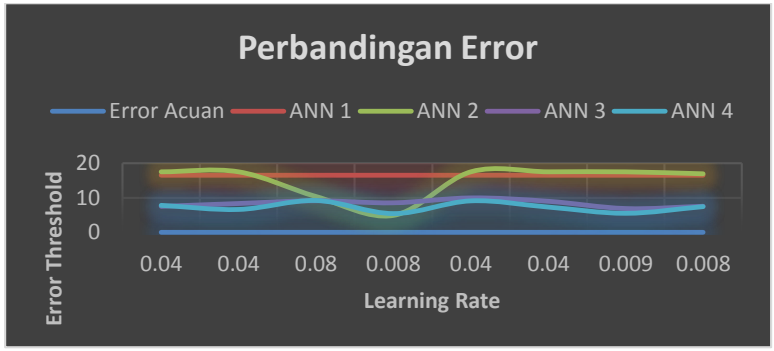
Jika dilihat dari 8 kali percobaan dengan konfigurasi yang berbeda, dapat dilihat bahwa ANN 1 dan ANN 2 tidak pernah menyentuh MSE, sehingga training dilakukan hingga epoch habis. Hasilnya, bobot jaringan pada ANN 1 dan ANN 2 tidak optimal. Beberapa hal yang menyebabkannya diantaranya nilai-nilai data yang masuk ke ANN 1 dan ANN 2 memiliki nilai kemiripan yang cukup mirip, namun memiliki kunci tebakan yang berbeda.

Tabel 4. 2 Contoh data latih dalam bentuk *.csv

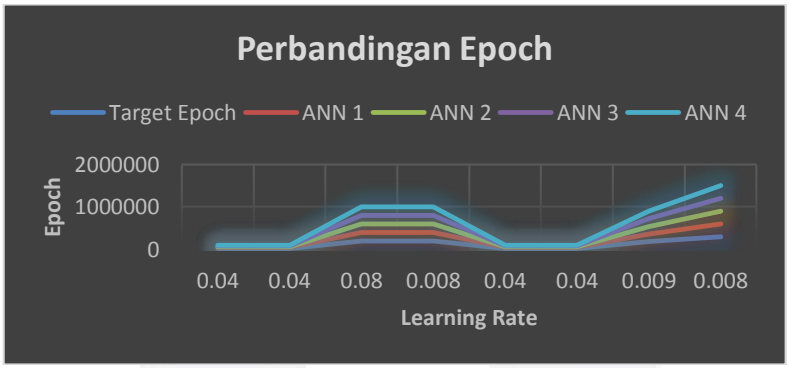
Dist 1	Alpha 1	Beta 1	Dist 2	Alpha 2	Beta 2	Dist 3	Alpha 3	Beta 3	Dist 4	Alpha 4	Beta4	Outcome
575	161	163.183	553	34.237	81.021	559	17.947	24.501	549	100.102	114.036	180
553	34.611	64.378	555	158.453	163.96	566	19.061	24.425	548	99.682	114.409	170
553	34.801	82.705	554	118.855	160.988	569	18.895	24.291	547	99.01	114.903	160
553	68.154	82.695	550	150.513	158.486	533	18.138	65.858	548	120.877	112.961	150
549	64.163	98.484	540	130.799	111.949	527	80.028	78.041	0	0	0	140
502	67.071	78.868	523	18.9	24.366	0	0	0	0	0	0	130
486	24.632	79.043	0	0	0	0	0	0	0	0	0	120
497	24.707	74.575	515	20.225	59.369	0	0	0	0	0	0	110
519	20.577	24.179	0	0	0	0	0	0	0	0	0	100
506	161.501	163.86	508	18.891	59.141	0	0	0	0	0	0	90
543	159.52	163.351	516	104.769	121.85	0	0	0	0	0	0	83
544	159.722	163.521	524	95.736	116.916	0	0	0	0	0	0	82.5
522	82.974	108.814	544	159.647	163.521	0	0	0	0	0	0	80
544	159.52	163.351	533	75.128	96.193	0	0	0	0	0	0	70
520	145.604	164.223	0	0	0	0	0	0	0	0	0	60
545	160.332	164.03	532	67.338	91.103	0	0	0	0	0	0	50
521	100.28	158.114	0	0	0	0	0	0	0	0	0	40
544	159.669	163.351	525	81.365	78.924	0	0	0	0	0	0	30
677	86.593	101.425	555	39.289	80.637	648	20.845	70.291	578	106.397	120.926	20
554	34.324	82.004	565	159.47	163.69	533	18.08	66.34	568	105.481	120.136	10
552	120.634	164.03	522	17.103	29.31	0	0	0	0	0	0	1

Dari hasil percobaan yang dilakukan 8 kali, penulis dapat mengambil kesimpulan bahwa setiap parameter seperti error threshold, epoch dan learning rate dapat memperngaruhi hasil training dari setiap ANN. Pada kasus ini, penggunaan hidden layer tidaklah efektif jika dilihat dari Tabel 4.1. Dari hasil grafik percobaan, ANN 2 memiliki tingkat error yang cukup tinggi yaitu 17.51074, jauh dari error threshold yang telah ditentukan. Hal ini dapat disebabkan oleh range nilai data yang masuk ke ANN 2 kurang variatif, dan juga karena salah satu

objek untuk pelatihan memiliki permukaan mengkilat, sehingga pemantulan infrared untuk mengidentifikasi nilai jarak dan sudut tidaklah sempurna.
Error yang cukup tinggi juga mempengaruhi epoch akhir dari training. ANN 2 selalu menyentuh epoch terakhir dalam training dikarenakan ANN 2 tidak dapat menyentuh error threshold yang telah ditentukan. Sehingga ANN 2 hanya akan berhenti training jika telah menyentuh epoch terakhir. Kondisi demikian juga terjadi pada ANN yang lainnya



Gambar 4. 6 Grafik Perbandingan Error



Gambar 4. 7 Grafik perbandingan training yang menyentuh target epoch

Berikut Tabel pengujian 3 dari penelitian

No	Masukan	Keluaran	Keputusan	akurasi
1	90	100	Benar	90%
2	80	100	Salah	80%
3	80	100	Salah	80%
4	70	100	Salah	70%
5	70	100	Salah	70%
6	60	100	Salah	60%
7	60	100	Salah	60%
8	50	100	Salah	50%
9	50	100	Salah	50%
10	40	100	Salah	40%
11	40	100	Salah	40%
12	30	100	Salah	30%
13	30	100	Salah	30%
14	20	100	Salah	20%
15	20	100	Salah	20%
16	10	100	Salah	10%
17	10	100	Salah	10%
18	10	100	Salah	10%
19	180	100	Salah	56%
20	180	100	Salah	56%
21	170	100	Salah	59%
22	170	100	Salah	59%
23	160	100	Salah	63%
24	160	100	Salah	63%
25	150	100	Salah	67%
26	150	100	Salah	67%
27	140	100	Salah	71%
28	140	100	Salah	71%
29	120	100	Benar	83%
30	120	100	Benar	83%
31	120	100	Benar	83%
32	110	100	Benar	91%
33	100	100	Benar	100%
34	100	100	Benar	100%
35	30	110	Salah	27%
36	30	110	Salah	27%
37	160	180	Benar	89%
38	160	110	Salah	69%
39	20	160	Salah	13%
40	180	180	Benar	100%
41	180	180	Benar	100%
42	1	180	Salah	1%
43	1	180	Salah	1%
44	83	110	Salah	75%
45	84	110	Salah	76%
46	84	110	Salah	76%
47	80	110	Salah	73%
48	80	110	Salah	73%
49	80	80	Benar	100%
50	70	110	Salah	64%
51	50	110	Salah	45%
52	30	160	Salah	19%
53	20	180	Salah	11%
54	1	180	Salah	1%
55	1	160	Salah	1%
56	90	180	Salah	50%
57	90	150	Salah	60%
58	110	110	Benar	100%
59	130	150	Benar	87%
60	140	180	Salah	78%
61	150	150	Benar	100%
62	150	150	Benar	100%
63	160	180	Benar	89%
64	160	150	Benar	94%
65	170	180	Benar	94%
66	170	180	Benar	94%
67	170	170	Benar	100%
68	180	170	Benar	94%
69	180	170	Benar	94%
70	90	170	Salah	53%
71	180	140	Salah	78%
72	20	140	Salah	14%
73	150	140	Benar	93%

74	140	135	Benar	96%
75	140	140	Benar	100%
76	140	140	Benar	100%
77	140	135	Benar	96%
78	10	140	Salah	7%
79	10	135	Salah	7%
80	10	10	Benar	100%
81	135	140	Benar	96%
82	135	135	Benar	100%
83	140	140	Benar	100%
84	140	140	Benar	100%
85	140	140	Benar	100%
86	150	140	Benar	93%
87	150	140	Benar	93%
88	160	140	Benar	88%
89	170	140	Benar	82%
90	170	140	Benar	82%
91	170	140	Benar	82%
92	180	140	Salah	78%
93	20	150	Salah	13%
94	20	150	Salah	13%
95	170	170	Benar	100%
96	170	170	Benar	100%
97	170	180	Benar	94%
98	170	160	Benar	94%
99	170	180	Benar	94%
100	180	180	Benar	100%
101	180	180	Benar	100%
102	10	150	Salah	7%
103	10	150	Salah	7%
104	150	150	Benar	100%
105	150	150	Benar	100%
106	150	150	Benar	100%
107	150	150	Benar	100%
108	150	150	Benar	100%
109	160	150	Benar	94%
110	160	180	Benar	89%
111	160	150	Benar	94%
112	170	170	Benar	100%
113	170	170	Benar	100%
114	170	180	Benar	94%
115	180	170	Benar	94%
116	180	180	Benar	100%
			Akurasi	70%

5. Kesimpulan

Berdasarkan hasil penelitian dan pembahasan yang telah dilakukan, dapat disimpulkan bahwa:

1. Arsitektur JST menghasilkan nilai MSE paling minimum dengan arsitektur yang terdiri dari 3-6-9-12 *input layer* yang dapat menyesusaikan berdasarkan jumlah objek yang masuk ke *pre input*, 0 *hidden layer* karena penggunaan *hidden layer* pada kondisi yang di sesuaikan penulis tidak dapat menghasilkan nilai MSE yang sesuai, dan 1 *ouput layer* karena sistem yang dirancang penulis dirancang untuk menghasilkan 1 keputusan berbelok untuk robot yang bernilai antara 0-180 derajat.
2. Variabel yang sangat mempengaruhi akurasi JST adalah *learning rate*, *error threshold*, dan data latih dari sensor Kinect. Karena pada dasarnya jaringan syaraf tiruan merupakan perhitungan matematis dimana hasilnya sangat dipengaruhi oleh nilai-nilai variabel pembentuk algoritma jaringan syaraf tiruan itu sendiri.
3. Ambang batas yang baik pada proses pelatihan agar dapat mengenali konfigurasi objek pada lingkungan yang telah disesuaikan adalah saat *Error threshold* = 0,01; *Total Epoch* = 200000, 20000, dan 180000; *Learning rate* = 0.08, 0.008, 0.04, dan 0.009; *Hidden layer* = 0. Dan dengan kondisi konfigurasi demikian, jaringan syaraf tiruan *multilayer feedforward* dapat diimplementasikan pada robot dan robot dapat bergerak secara efisien dalam menghindari objek diam.

Saran

Untuk pengembangan tugas akhir ini dimasa yang akan datang, ada beberapa saran yang perlu diperhatikan yaitu sebagai berikut:

1. Mengembangkan Aplikasi kecerdasan buatan untuk mengenali konfigurasi lintasan dengan tanpa adanya batasan objek yang dideteksi.
2. Mengembangkan Aplikasi kecerdasan buatan untuk mengenali dan menghindari objek bergerak.
3. Mengembangkan Aplikasi kecerdasan buatan untuk mengenali konfigurasi lintasan dengan menggunakan sensor yang lebih efektif dan memiliki komputasi yang sangat cepat agar dalam pengambilan data training dapat dilakukan dengan cepat dan saat pengujian keluaran gambar pada GUI tidak lag.

Daftar Pustaka

[1] Kolski, S., & friends. (2006). Autonomous Driving in Structured and Unstructured Environments. Intelligent Vehicles Symposium, 14-4, 558-563.

[2] Correa, D. S., & friends. (2012). Mobile Robots Navigation in Indoor Environments Using Kinect Sensor. Second Brazilian Conference on Critical Embedded Systems, 36-41.

[3] Indrawan, G. (2008). Perancangan dan Implementasi Kecerdasan Buatan Robot Pencari Jalur berbasis Mikrokontroler Basic Stamp. Depok: Fakultas Teknik, Universitas Indonesia.

[4] Ardisasmita, M. S. (2003). Pengembangan Robot Mobil Otonom Menggunakan Sistem Kendali Fuzzy dan Jaringan Syaraf Tiruan. Risalah Lokakarya Komputasi dalam Sains dan Teknologi Nuklir XIV, 157-170.

[5] Kusumadewi, S. (2004). Membangun Jaringan Syaraf Tiruan (menggunakan MATLAB & Excel Link). Yogyakarta: Graha Ilmu.

[6] Pitowarno, E. (2008). Serial Buku Robotik: Kecerdasan Buatan. Yogyakarta: Andi Publisher.

[7] Purnomo, M. H., & Kurniawan, A. (2006). Supervised Neural Networks. Yogyakarta: Graha Ilmu.

[8] Viager, M. (2011). Analysis of Kinect for Mobile Robots. Kongens Lyngby, Denmark: Technical University of Denmark.